
Subject: Re: Reducing an array.
Posted by [Craig Markwardt](#) on Mon, 30 Sep 2002 23:17:39 GMT
[View Forum Message](#) <> [Reply to Message](#)

"Joe" <foosej@hotmail.com> writes:

> Hi- I'm somewhat new to IDL and was wondering what the most efficient way
> is to 'OR' all the elements of an array together resulting in a scalar
> value. I'm hoping IDL has a built-in way of doing this rather than using a
> FOR-LOOP. Similar to how IDL has the TOTAL function which sums all the
> elements of an array together. I've used other languages which allow you
> to 'reduce' arrays to a scalar using an arbitrary function (i.e. Python's
> reduce function).
>
> What I am doing is taking a lot of integer data which is either 0's or 1's
> and compressing it into the bits of 64-bit unsigned integers. Here is a bit
> of sample code:
>
> data = [1,0,0,0,1,1,1,0,1,0,1,0,0, ... , 0, 1, 0, 1] ; bunch of data, assume
> # of elements is multiple of 64
> shifts = reverse(indgen(n_elements(data))) MOD 64
> compressed_data = ishft(data,shifts)
> ; here is where I want to take the compressed_data array and make it into a
> ; bunch (n_elements(data)/64, to be exact) of unsigned 64-bit integers by
> OR'ing
> ; every 64 elements of compressed data together

In this case you can use TOTAL() directly. First you REFORM() your data into a 2-d array, 64xN, then then total the 1st dimension. This works because each of your values has only one data bit set, so summing and ORing are equivalent.

```
compressed_data = reform(compressed_data, 64, n_elements(compressed_data)/64)
result = total(compressed_data, 1)
```

That's it! For JD, I could have combined both statements onto one line, but this is more readable.

Craig

--

Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
